|  |
| --- |
| University of missouri-kansas city |
| SternerLearn |
| Project Proposal & Plan |
|  |
| **Devin Turner & Connor Ledgerwood** |
| **2/22/2013** |

|  |
| --- |
|  |

# Introduction

There is no question that today's children are facing more distractions than ever to keep them unfocused and unwilling to learn. The ever-increasing prevalence of mobile devices continue to improve the world's access to the vast expanse that is the internet, but in some cases this may not be a benefit. The nature of instant gratification and communication over the web constantly tempts students who should be studying. To combat these cases in which the student is too stubborn or sidetracked to study, a mobile application called SternerLearn is proposed. This program will serve as a means to convey important information to parents and students. But, if parental knowledge of curriculum is not enough to ensure good behavior and timely completion of assignments, SternerLearn can act as a unique, configurable, web-based enforcement tool for parents until their child has improved on his or her educational goals.

# Project Goal and Objectives

## Overall Goal

The project aim is to create a mobile interface for parents to monitor their child’s progress in school. Stemming from an apparent disconnect of communication between parents, students, and instructors in many schools, this tool will act to provide cohesion needed between the three when the student is not a willing relay of scholastic information. Parents will be provided with instant access to the performance of their child, the details of what work is expected for each class, and whether or not deadlines have been met. In addition to these features, the student will have access to assignment-specific data and teaching resources to aid them in their studies. Finally, a digital disciplinary measure controlled by the parent will allow for both the enactment of repercussions to be placed on the student and enhanced monitoring features by the parent.

## Specific Objectives

This project's objective is to create a secure mobile app that can be used by parents to help reinforce the value of placing an interest in their child's studies. The app will have access to access a server containing thorough scholastic information uploaded by teachers and staff of the school system. The visual interface will provide clear information in a way that is beneficial to the user depending on whether they are parent or student. The application will be able to provide system notifications by interfacing with the mobile API to provide users with a greater awareness of urgent educational information. Ultimately, it will be able to diagnose if the student is failing to meet syllabus criteria or adhere to school policies, reporting to the parent and imposing penalties that they have set.

## Significance

This app will improve the quality of learning at participating school districts. Countless frustrations have risen because no clear channel of correspondence exists between the home and the school. Since an internet terminal now resides in the pockets of most, creating such a channel properly could come with extreme benefit to the district. School funding is in part judged by the effectiveness of producing quality GPA, and this app can address students who would have little interest in maintaining grades to push towards a better looking district and an increase in educational finances. Apart from providing the central functionality of a virtual report card and calendar, this app would expose the inner mechanisms of the classroom to otherwise uninformed parents. Essentially, teachers would be able to construct an organized learning environment with a valid lecture plan so that parents are not only satisfied with their child’s intellectual growth but also in the nature of his or her studies. This type of open resource could be very healthy to schools by improving the quality of both students and instructors.

# Project Background

Many other platforms exist with similar features to SternerLearn, but none of them can provide the full spectrum of education, communication, and parental control. Below are several popular programs and applications that support features included in this project.

**Blackboard**

Blackboard is primarily a browser-based implementation although a mobile version is available on limited service providers. It is implemented by entire schools to act as an interface between students and their instructors. It features a website for each enrolled class that may be linked to any uploaded file as well as a system for submitting files to instructors electronically. It can also post reminders of due dates. However, it is only designed to be used by responsible students.

1. **Dash**

Dash acts as a mobile communication interface between parents and teachers. Unlike blackboard, it is mainly intended for younger students. The app allows for a quick summary of the behavior of children to be sent from a teacher's mobile device to the guardian's and integrates with the Android platform to provide texting and calling. It lacks the features pertaining to grades and planning.

1. **HomeWork**

HomeWork is not a communication tool at all but an Android app designed to track school deadlines into a mobile calendar. This app has multiple features to enhance planning such as creating classes with stored contact information and assignments with stored properties and also integrates with the Google calendar. But, it is not service-oriented.

1. **PowerSchool**

This web-based system contains a wide array of features. It is a centralized web-based program that uses databases to manage student and faculty information, class schedules and assignments, attendance and discipline tracking, a scheduling tool, and report generation. These are many of the features that SternerLearn provides, but it does not give parents any direct control of the learning environment.

1. **Edmodo**

Edmodo is another centralized learning tool connecting teachers and students that has begun to migrate to mobile platforms. This progam provides a method to quickly and directly communicate with instructors, who can also use the tool as a grade book. It's biggest feature is the ability to dynamically network with anyone wishing to get involved with the school district.

1. **eBlaster Mobile**

Eblaster Mobile is a step in another direction. Since SternerLearn requires monitoring and access control capabilities, this app is worthy of consideration. Basically, it functions as a background application on Android devices that is able to log all activity on the device, from text messaging to emails to voice call details.

1. **MMGuardian**

Whereas eBlaster Mobile is solely an activity logger, MMGuardian allows for direct parental control to be placed on an Android device. It has a GPS locator of the phone, may set restrictions on usage based on time of day, and can control when applications can be used.

# Proposed System

## Requirement Specification

### Functional/Business Requirements

Requirements are each given a priority of 1-10, with 10 being the highest priority.

**Mobile Interface**

* (10) The system shall have an interface accessible through an application available on mobile Android platforms.
* (10) The mobile interface shall have 3 account options: guardian, teacher, and student.
* (10) Guardian accounts shall be able to read school data, modify parental controls, and read student tracking.
* (10) Teachers shall be able to read and write school data.
* (10) Students shall be able to read school data.

**Web Interface**

* (10) The system shall have an interface accessible through any common web browser.
* (10) The web interface shall have the same capabilities as the mobile interface.

**School Data**

* (10) School data shall contain grades, assignments, tests, classes, attendance, students, and guardians.
* (10) Grades shall be correlated with assignments and tests.
* (10) Assignments shall have a grade and a due date.
* (10) Tests shall have a grade and a date.
* (10) Classes shall have a teacher and a list of students.
* (10) Students shall have a list of classes and a list of guardians and their attendance record.

**Notifications**

* (9) The system shall have the ability to send emails to guardians.
* (9) Emails shall automatically be sent to guardians when students fail assignments or tests, are tardy, or absent.
* (6) Guardians shall be able to configure or disable the automatic email system.
* (6) Teachers shall be able to manually send an email to a given student’s guardians.

**Parental Control**

* (10) Guardians shall be able to view and configure settings on their student’s Android device if the student also has the mobile application installed.
* (10) Guardians shall be able to view their student’s past and current location using the Student Tracking.
* (8) Guardians shall be able to view their student’s text messages.
* (6) Guardians shall be able to disable applications for given time frames.
* (10) Student academic performance shall be rated by a tiered system of based on
* (8) The guardian’s application shall be granted access to the student’s device at preset behavior levels.
* (7) These levels shall be configurable by the guardian.
* (8) The preset levels shall have defaults depending on the student’s grades, absences, and tardiness.
* (10) The student’s application shall have a secondary password to prevent uninstallation of the application without the guardian’s consent.
* (10) The system shall automatically record the longitude and latitude coordinates of the student’s device every 15 minutes.
* (10) The coordinates shall be stored for future access.

### Non-functional/Technical Requirements

* (10) All persistent data shall be stored in SQL databases on an IBM cloud machine.
* (10) Data tables shall store data on each of the following: schools, staff members, user accounts, students, courses, and assignments.
* (10) Data tables shall store student-specific data related to his or her academics as well as data related to parent-controlled features such as detentions, missed due dates, and GPS tracking.
* (10) The response time shall be low enough to not affect the user experience detrimentally.
* (9) The system shall securely communicate personal information between the server and clients.
* (6) The system shall locally store messages to the server while disconnected from the server.
* (5) The system shall cache data from the server for viewing while disconnected from the server.

### Business Process/Workflow analysis:

Our group plans to work in an agile fashion to produce rapid development. We will divide tasks as appropriate to group member’s areas of skill and expertise, as well those task’s relations and dependencies to other tasks in the project. The main focus will be development, but as development occurs so will bugs. In our workflow we will guarantee minimal amounts of bugs by responding to bugs immediately. Instead of simply noting the bug and moving on with development, that bug will be prioritized. Too often bugs fixes are postponed to the end of a project, and then the project runs out of time and gets rushed out the door full of bugs. To avoid this, we will test thoroughly and fix bugs as they arise.

### Technological and Architectural requirements

* (10) The mobile interface shall require an Android operating system.
* (10) The Android application shall be designed using JQuery Mobile.
* (10) The web interface shall require a web browser.
* (10) All interfaces shall require an internet connection to fully function.
* (10) The data storage shall require SQL.
* (10) The server shall require the .NET framework.

## Framework Specification

### Assumptions and Principles

Our framework will be based on a service-oriented architecture. Most of the logic and data will be stored on a server. Ideally it would be multiple servers to distribute the load and handle peak load, but to simplify matters we are using a single server. This server will store the databases which our web services use to provide data to any clients. Clients will be running the mobile application, which when connected to the internet can access our web services and other existing services to retrieve data it needs to display to the user. We assume that both the server and the mobile device have a reliable internet connection.

### System Architecture Diagram



## System Specification

### Existing Services

We do not current plan on using any existing services. Our project is relatively self-contained and so the services we plan to create, as described below, should be enough for our requirements. However, as needs arise, we will investigate external services.

### New Services

#### AccountManagement

##### Class Diagram



##### Scenario and Use Case Specification Template

The first sequence diagram illustrates the user's initial function calls to the web service in order to sign up for a new SternerLearn account. After the web service checks the user's credentials for validity and compares them with existing entries in the database, an account is added to the database with unique ID numbers which are used to later query any stored information regarded to the account and its users. After the creation is successful, the user logs in to display the home page.

Sequence Diagram



##### Service Specification

This service contains all client facing functionality for the account tables of the database. This service will allow creation of new users of all types, and will authenticate the login of existing users.

Login(username, password)

The Login service takes two strings, the username and password, and determines whether it is a user, and if so, of what type (student, teacher, or guardian). It will return an integer corresponding to the validity and/or type of the user. This service is of the highest priority to allow users to begin interacting with the system. It should not be difficult to create.

Register(account data)

The Register service takes all of the requisite data to create a new account, including name, username, password, email address, and child or parent account information, depending on who is creating the account. It will return a Boolean to indicate success or failure. This service is of the highest priority to allow users to begin interacting with the system. This service will not be difficult to create, but will have to deal with a large amount of data.

#### StudentData

##### Class Diagram



##### Scenario and Use Case Specification Template

A student user can use the mobile interface to display information about enrolled courses and other assignment-specific data. This process simply consists of database database queries to corresponding tables given a student ID and other identifiers stored in the app as classes and assignments are taken from the web service to display.

Sequence Diagram



An instructor or other staff member can use the web service in much the same way as the student. However, a teacher will need to set grades rather than view them. This is why a log-in given a staff ID will allow for the user to create classes, assign existing students to classes, form assignments, and issue grades for assignments to each student.

Sequence Diagram



##### Service Specification

This service contains all client facing functionality for the school data tables of the database. This service’s read functions are intended to be used by all users, and the write functions are only available to teacher accounts. This is where the bulk of the functionality exists for the application. All of these services below are of the highest priority and are of medium difficulty. They should all be relatively similar, generating results by using the appropriate SQL commands on the data.

getClasses(studentID)

This takes an integer as the student ID and first determines if that is a valid student. If so, it returns a list of classes the student is enrolled in. If not, it returns an error code.

getGrades(studentID, classID)

This takes an integer as the student ID and another integer as the class ID and first determines if it has valid student and class IDs. If so, it returns a list of the assignments for that class, and the grades the student received on them. If not, it returns an error code.

getInfractions(studentID)

This takes an integer as the student ID and first determines if that is a valid student. If so, it returns a list of infractions the student received, including dates and descriptions. If not, it returns an error code.

getParentID(studentID)

This takes an integer as the student ID and first determines if that is a valid student. If so, it returns the parent ID associated with that account. If not, it returns an error code.

getStudentID(parentID)

This takes an integer as the parent ID and first determines if that is a valid parent. If so, it returns the student ID associated with that account. If not, it returns an error code.

addClass(teacherID, name)

This takes an integer as the teacher ID and the name of the class. It first determines if that is a valid teacher. If so, it adds the class with a new class ID and returns a success code. If not, it returns an error code.

addAssignment(teacherID, name, points, test, dateDue, classID)

This takes an integer as the teacher ID and all of the assignment data. It first determines if that is a valid teacher. If so, it adds the assignment with a new assignment ID and returns a success code. If not, it returns an error code.

addStudentToClass(teacherID, studentID, classID)

This takes an integer as the teacher ID, an integer as the student ID, and an integer as the class ID. It first determines if those are all valid IDs. If so, it adds the student to the class and returns a success code. If not, it returns an error code.

addInfaction(teacherID, infractionType, studentID, dateTime)

This takes an integer as the teacher ID and infraction data. It first determines if that is a valid teacher. If so, it adds the infraction with a new infraction ID and returns a success code. If not, it returns an error code.

addGuardian(teacherID, studentID, guardianID)

This takes an integer as the teacher ID, an integer as the student ID, and an integer as the guardian ID. It first determines if those are all valid accounts, that the student has a remaining link, and the guardian is not linked. If so, it adds the guardianID to the student’s linked accounts and returns a success code. If not, it returns an error code.

addGrade(teacherID, studentID, assignmentID, pointsReceived)

This takes an integer as the teacher ID, an integer for the student ID, and the grade data. It first determines if that is a valid teacher, student, and assignment. If so, it adds the grade and returns a success code. If not, it returns an error code.

#### ParentalManagement

##### Class Diagram



##### Scenario and Use Case Specification Template

The parent-side sequence diagram shows unique features available to them. In this scenario, GPS tracking information is periodically logged by the student app. When the parent checks their child's location, perhaps the resulting data is unpleasing. The parent then adds their own infraction and enables a new restriction in the options menu. This results in the student recognizing its enactment and using the web service to query information regarding the infraction placed upon him or her.

Sequence Diagram



##### Service Specification

This web service will contain all client-facing functionality to access the parental control data. All of these services will only be accessible by parents who are linked to the specified student account.

addLocation(parentID, studentID, time, location)

This takes a parent ID, student ID, time, and location. It first verifies that the parent and student exist and are linked. If so, it adds the specified time and location to the database and returns a success code. If not, it returns an error code.

This service is of high priority. We find the tracking mechanism to be the most interesting parental control feature and so are prioritizing it over the others. This will be of medium difficulty.

getLocations(parentID, studentID, startTime, endTime)

This takes a parent ID, student ID, start time, and end time. It first verifies that the parent and student exist and are linked. If so, it returns a list of locations and times that are within the threshold. If not, it returns an error code.

addTextMessage(parentID, studentID, message)

This takes a parent ID, student ID, and message. It first verifies that the parent and student exist and are linked. If so, it adds the specified message to the database and returns a success code. If not, it returns an error code.

This service is of medium priority. It is not an essential portion of the system, but to have a reasonable student behavior level system, there need to be multiple parental controls. This is of higher difficulty, because we must determine how to access the text messages on the unit and transfer them.

getTextMessages(parentID, studentID, startTime, endTime)

This takes a parent ID, student ID, start time, and end time. It first verifies that the parent and student exist and are linked. If so, it returns a list of the messages associated with that student in the specified time frame. If not, it returns an error code.

getApplications(parentID, studentID)

This takes a parent ID and student ID. It first verifies that the parent and student exist and are linked. If so, it returns a list of applications on the student’s phone, and their current state (enabled or disabled). If not, it returns an error code.

This service is of medium priority. It is not an essential portion of the system, but to have a reasonable student behavior level system, there need to be multiple parental controls. This is of the highest difficulty, because we are not absolutely sure this functionality is built into Android, and will require investigation.

disableApplication(parentID, studentID, applicationID)

This takes a parent ID, student ID, time, and application ID. It first verifies that the parent and student exist and are linked. If so, it sets the disable flag for that application for that student. If not, it returns an error code.

#### Database

The database is where all of the data will be stored that the above web services will read and write to. The UML describing the tables in the database is below.



## Plan by Services

The schedules are all contained in the Iteration excel spreadsheets. The individual tasks and assignees are in the Tasks workbook. Essentially, we first plan to create the database structure. Then create the web services which can read and write to the databases, along with some basic interfaces to visualize the contents of the databases and to allow teachers to add grades en masse. Then once all of the backend work is complete, we will create the application using jQuery Mobile, and simply call the appropriate web services to access the data to display.

All of the tasks in the Agilefant spreadsheets have time estimates which account for a detailed design, implementation, creation of unit tests, and thorough testing. We did not create separate tasks for design and implementation to reduce task bloat.

Our schedule was designed to minimize risk. We implement and thoroughly test the backend first so that we are confident our web services are functioning before beginning on the application. In this fashion, we can complete the application in one shot instead of designing the UI with stubs for functionality. This will allow the application to actually be tested as the UI is implemented, which helps connect the experience for the developer and enforces the need for testing.

# Internet Access

## Project Website

<http://170.224.169.57/aspnet_client/website/website/main.html>

## Agilefant

https://cloud.agilefant.org/evaluator102/login.jsp

To log in, use the username “professor” and password “password”.

## GitHub

<http://github.com/clkv5/cs551_project/>

# Bibliography

Blackboard. <http://www.blackboard.com/>

Dash. <http://dash4teachers.com/>

HomeWork. <https://play.google.com/store/apps/details?id=klwinkel.huiswerk&hl=en>

PowerSchool. <http://www.pearsonschoolsystems.com/products/powerschool/>

Edmodo. <http://www.edmodo.com/>

eBlaster Mobile. <http://www.eblaster.com/eblaster-mobile.html>

MMGuardian. <https://play.google.com/store/apps/details?id=com.mmguardian&hl=en>